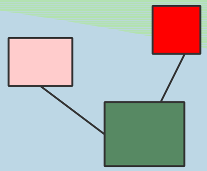
Common Component Architecture

http://www.cca-forum.org

Rob Armstrong
rob@sandia.gov

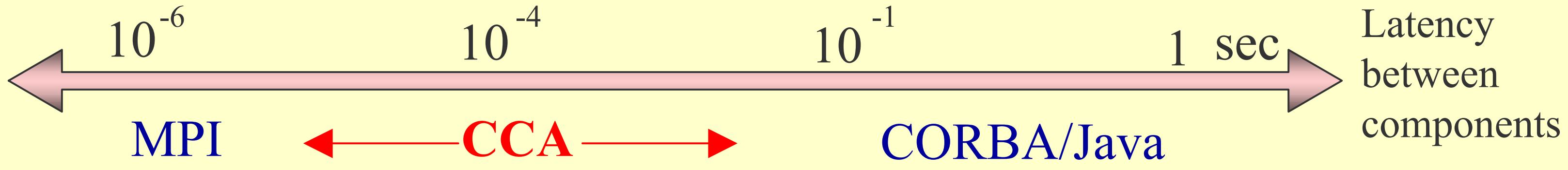
CCA Motivation

Desire to build Science Applications by hooking together drag-N-drop components.

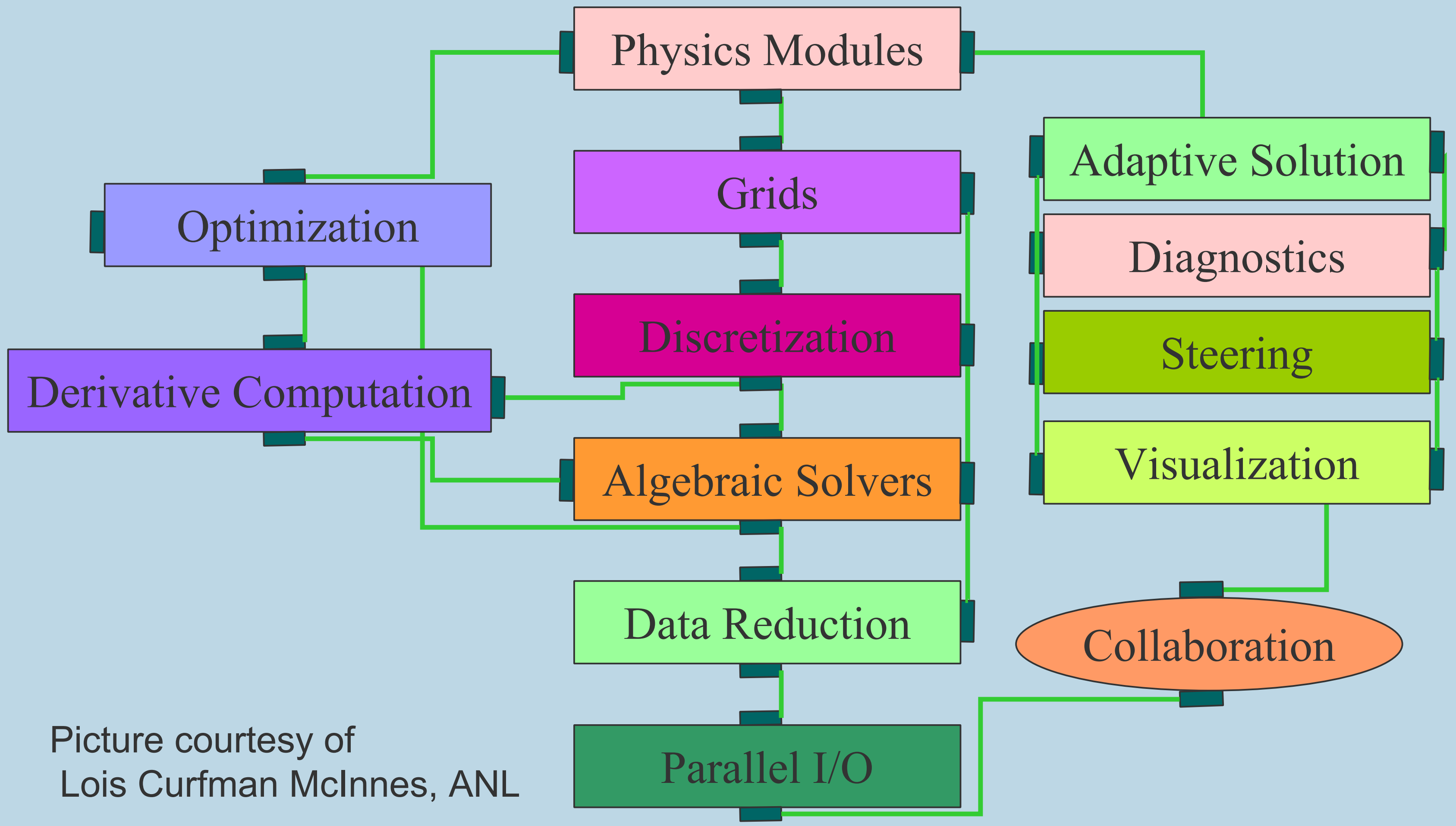


DOE Common Component Architecture provides a mechanism for interoperability of high performance components developed by many different groups in different languages or frameworks.

Existing Component Architecture Standards such as CORBA, Java Beans, and COM do not provide support for parallel components.



21st Century Application



Picture courtesy of
Lois Curfman McInnes, ANL

Who are we?

- Researchers in the HP components field that are dedicated to forming an open standard for HP components.
  - addressing the concerns of HPC.
- Originated from DOE□s DOE2K program
  - has its place in ASCI: Software Integration Curve.
  - participation from universities.

CCA Active Participants *

- Dennis Gannon (Indiana U)
- Randall Bramley (Cal Tech)
- Gary Kumfert, Scott Kohn, Tom Epperly (Livermore)
- Craig Rasmussen, Kate Keahey (Los Alamos)
- Rob Armstrong, Ben Allan, Jaideep Ray (Sandia)
- David Bernholdt, Jim Kohl (Oak Ridge)
- Lois McInnes, Paul Hovland, Lori Freitag (Argonne)
- Steve Parker (U Utah)
- Jarek Nieplocha (Pacific Northwest Labs)
- Robert Clay, Lee Taylor (Terascale)

* Send "subscribe cca-forum" to major-domo@z.ca.sandia.gov to join CCA mailing list

The Requirements for HP Component Architecture

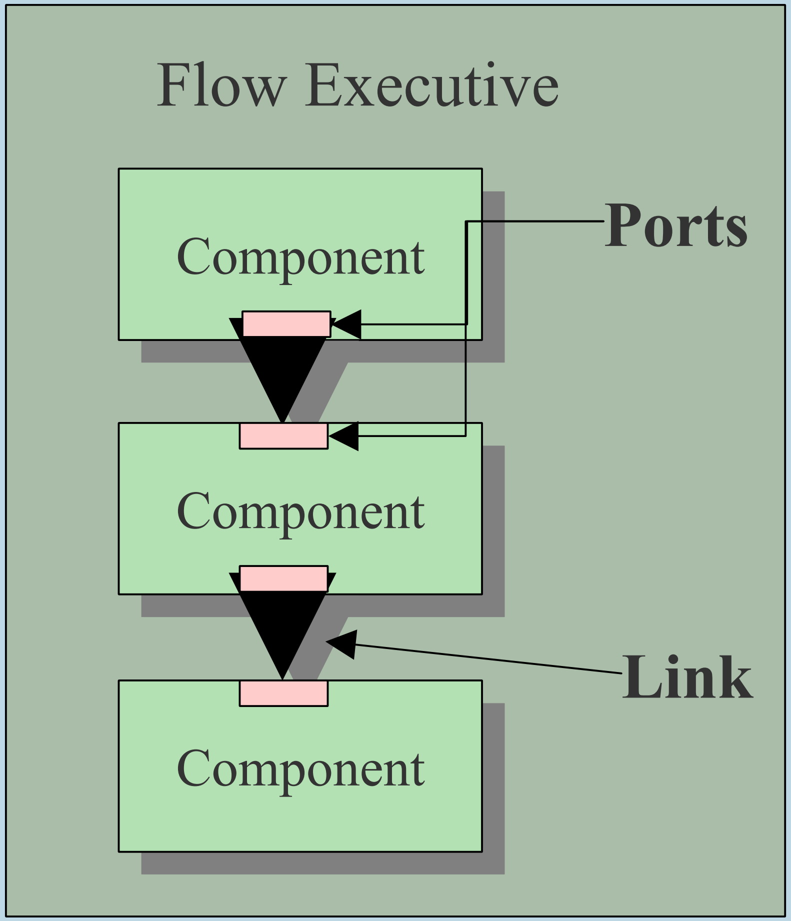
- Simple/Flexible
  - to adopt
  - to understand
  - to use
- Support a composition mechanism that does not impede high-performance component interactions.
- Permits the SPMD paradigm in component form.
- Meant to live with and rely on **other** commodity component frameworks to provide services ...
  - (e.g. JavaBeans, CCM, ...).

CCA is a component architecture that doesn't dictate frameworks or runtime

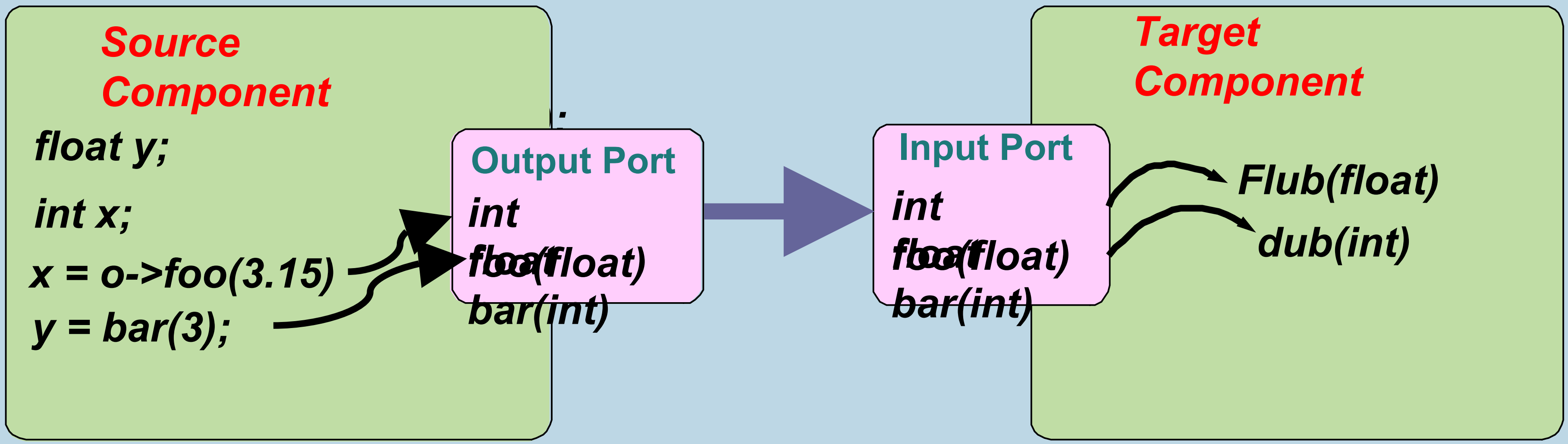
- We are trying to create components that are usable under a variety of frameworks;
- We are ***not*** making a new framework implementation.
    - provide a means for discovering interfaces, similar to *IUnknown.*
    - specifically exclude *how* the components are linked, that is the job of a framework.
    - provide language-independent means for creating components from existing and future languages.

What have we come up with?

- Scientific IDL spec.
  - provides language interoperability only - not part of the component arch. *per se.*
  - introspection.
- Metaphor from visual programming (e.g. AVS)
  - instead of data-flow, ports are linked with interfaces
  - Provides/Uses design pattern
  - 0.5 CCA specification: http://www.cca-forum.org



Port model hooks up an interface from one component to another



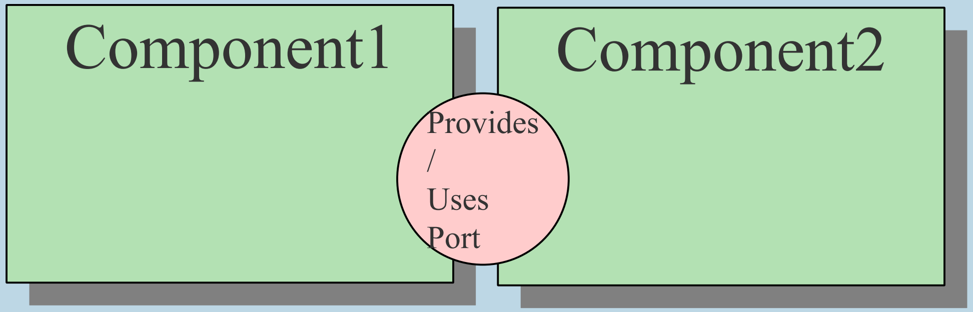
Data Flow Analogue:

Output Port == *Uses* Port

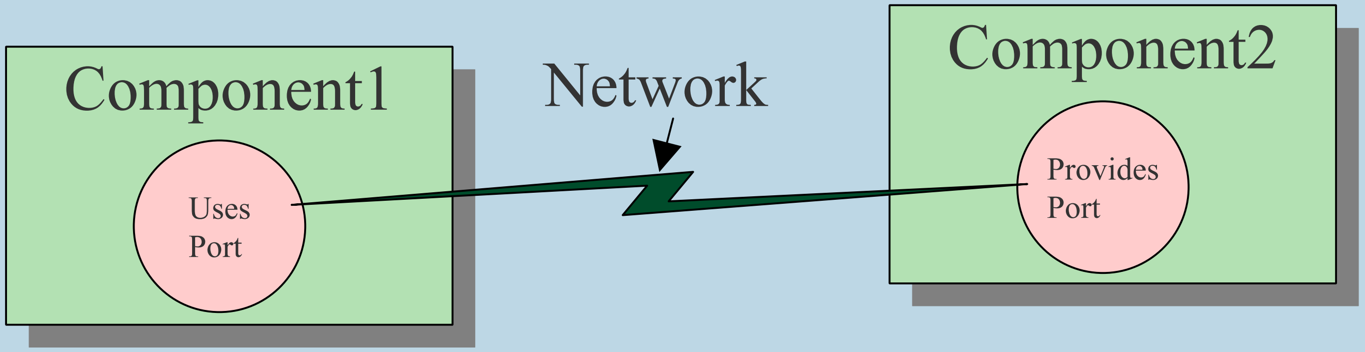
Input Port == ***Provides*** **Port**

Ports preserve the high-perf. of direct connections plus versatility of distributed object systems

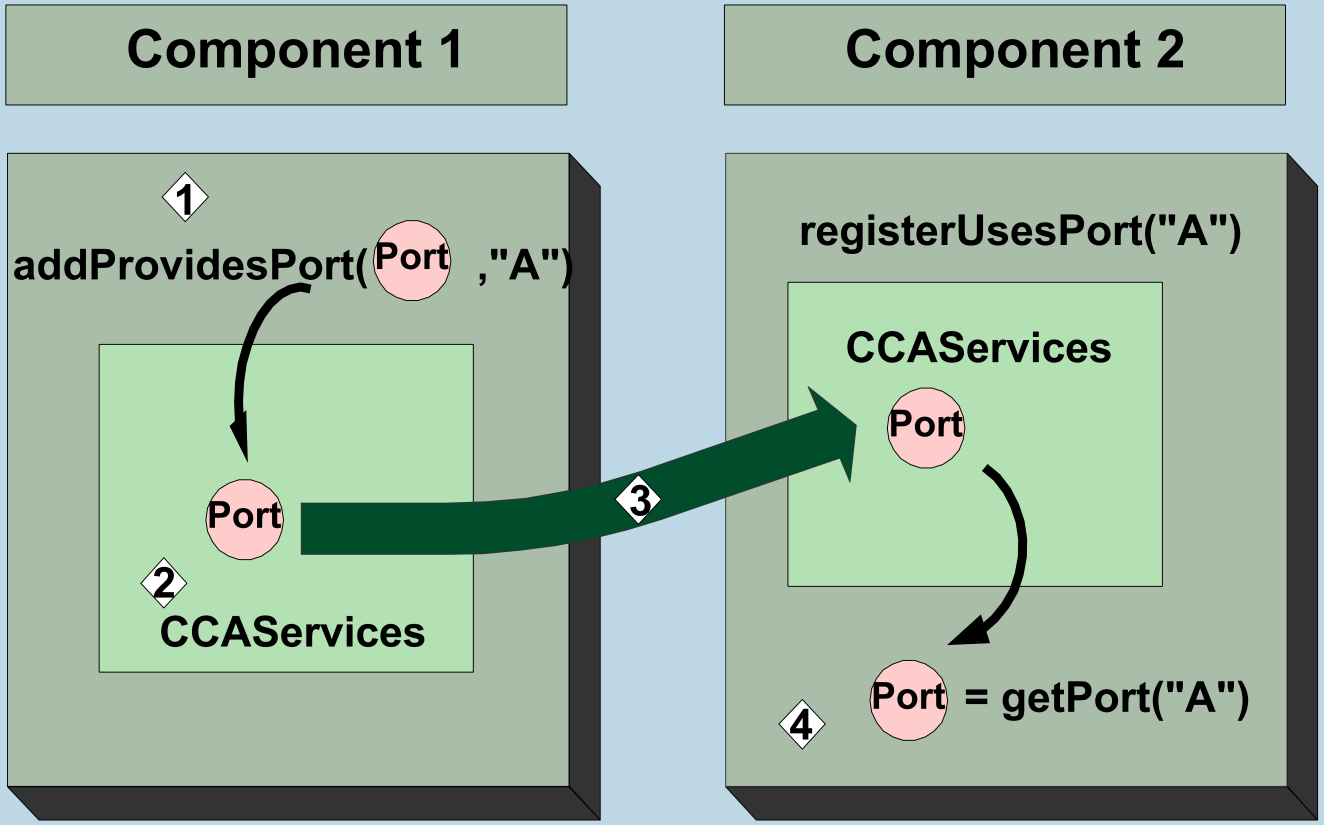
- Allows for directly connected interfaces: the next component is a few function calls away (w/ SIDL).



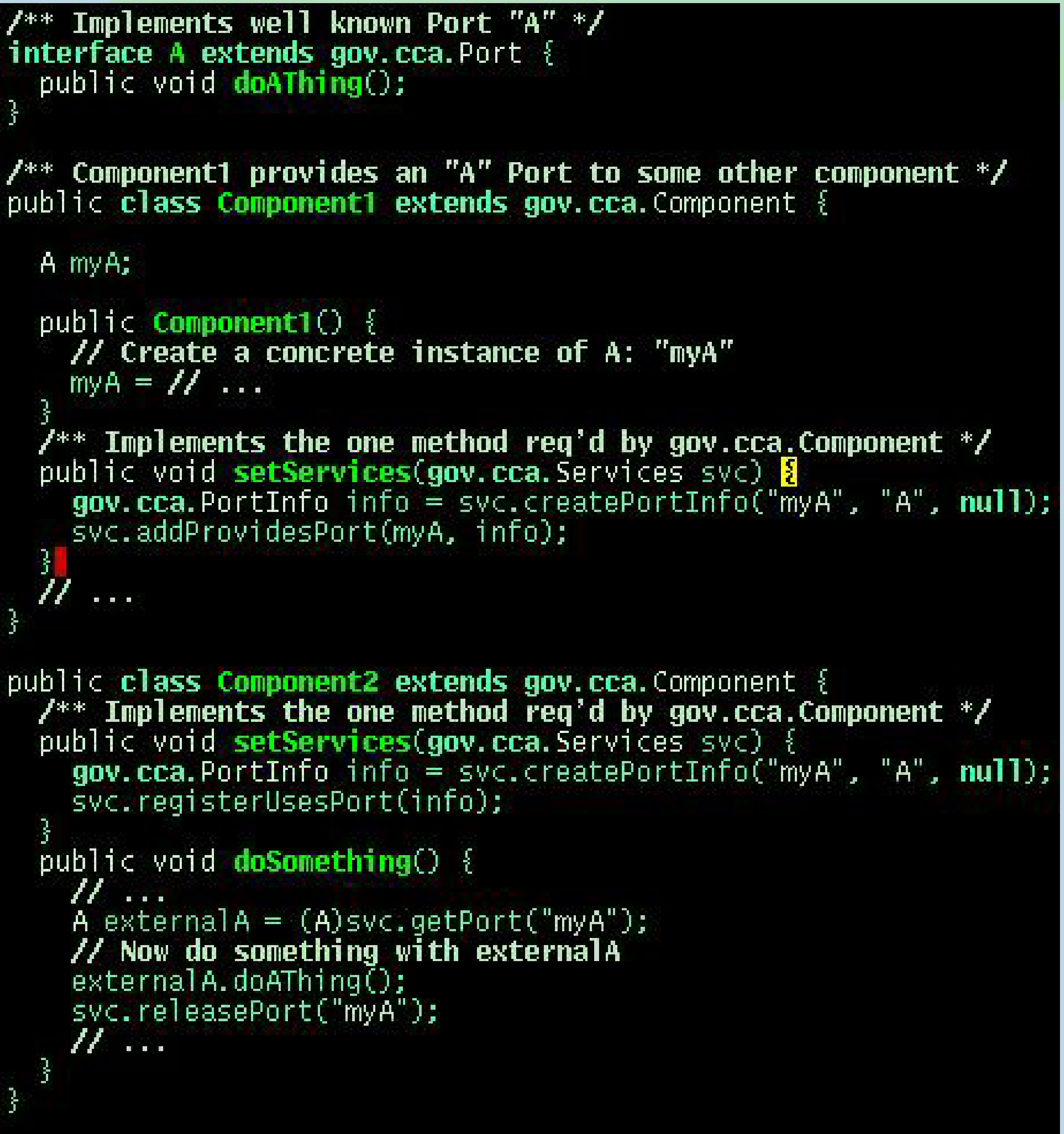
- Adapters will create network-distributed objects out of the same components without altering them.



Generalized ports and the provides/uses design pattern for coupling components



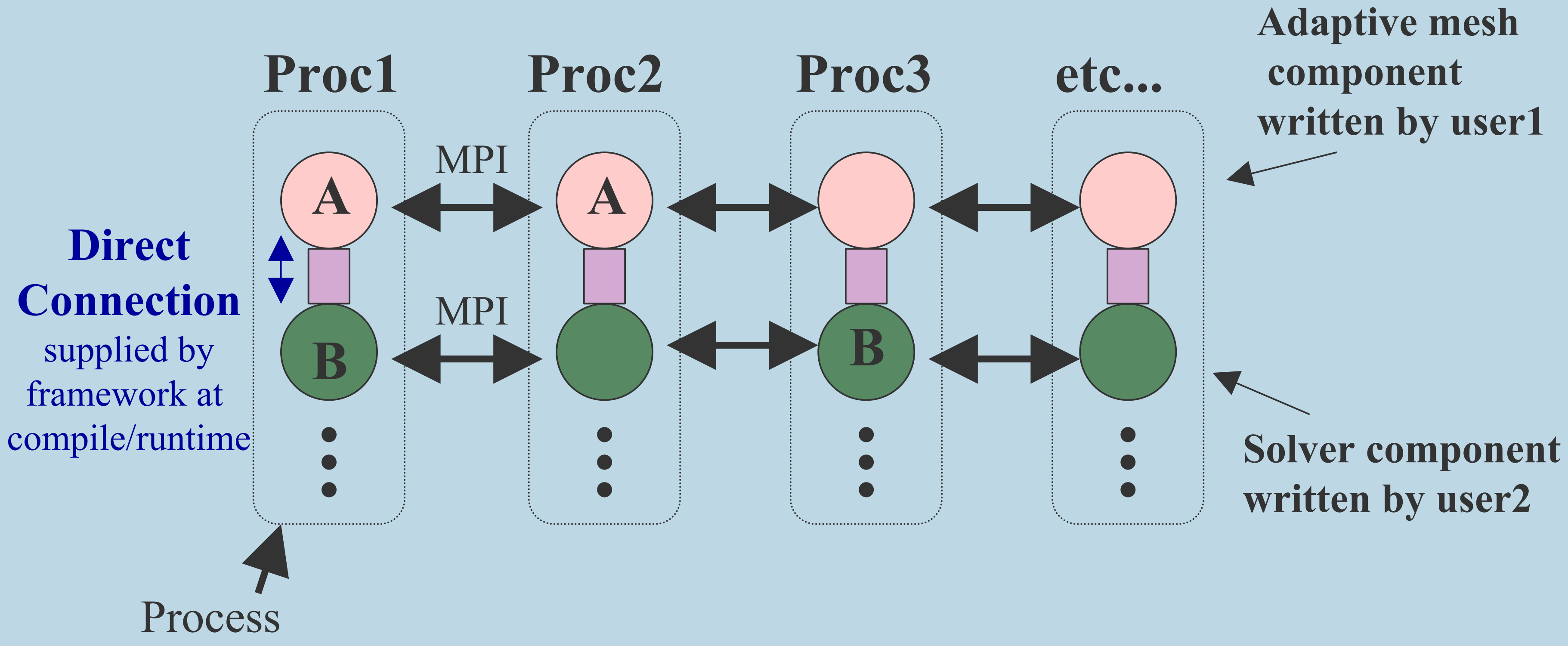
Example: connectable components



```
/** Implements well known Port "A" */
interface A extends gov.cca.Port {
  public void doAThing();
}

/** Component1 provides an "A" Port to some other component */
public class Component1 extends gov.cca.Component {

  A myA;

  public Component1() {
    // Create a concrete instance of A: "myA"
    myA = // ...
  }
  /** Implements the one method req'd by gov.cca.Component */
  public void setServices(gov.cca.Services svc) {
    gov.cca.PortInfo info = svc.createPortInfo("myA", "A", null);
    svc.addProvidesPort(myA, info);
  }
  // ...
}

public class Component2 extends gov.cca.Component {
  /** Implements the one method req'd by gov.cca.Component */
  public void setServices(gov.cca.Services svc) {
    gov.cca.PortInfo info = svc.createPortInfo("myA", "A", null);
    svc.registerUsesPort(info);
  }
  public void doSomething() {
    // ...
    A externalA = (A)svc.getPort("myA");
    // Now do something with externalA
    externalA.doAThing();
    svc.releasePort("myA");
    // ...
  }
}
```

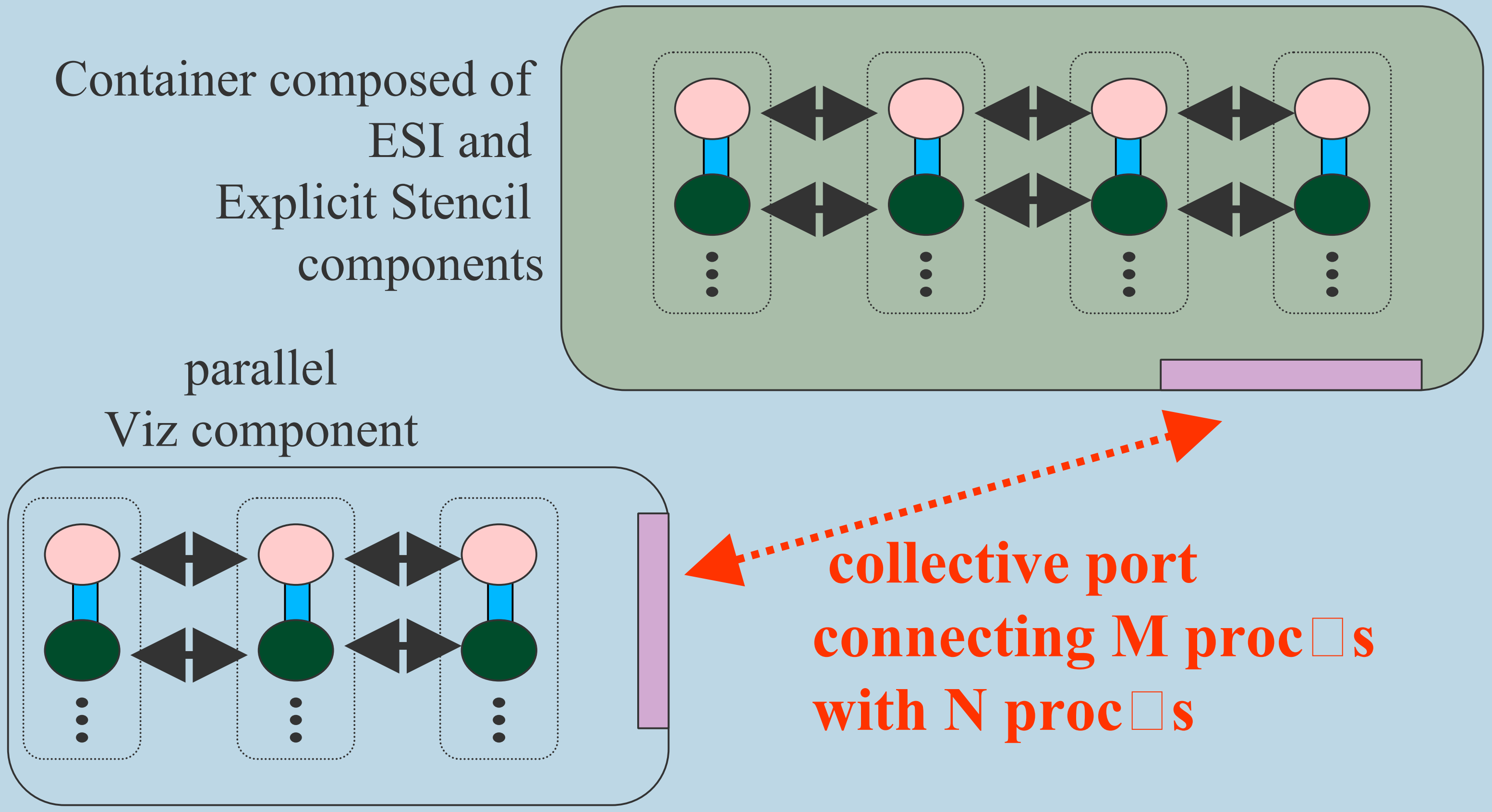
CCA concept of SCMD (SPMD HPC) components

MPI application using CCA for interaction between components A and B within the same address space


Proc1
Proc2
Proc3
etc...
Adaptive mesh component written by user1
MPI
A
A
MPI
B
B
Direct Connection
supplied by framework at compile/runtime
Solver component written by user2
Process

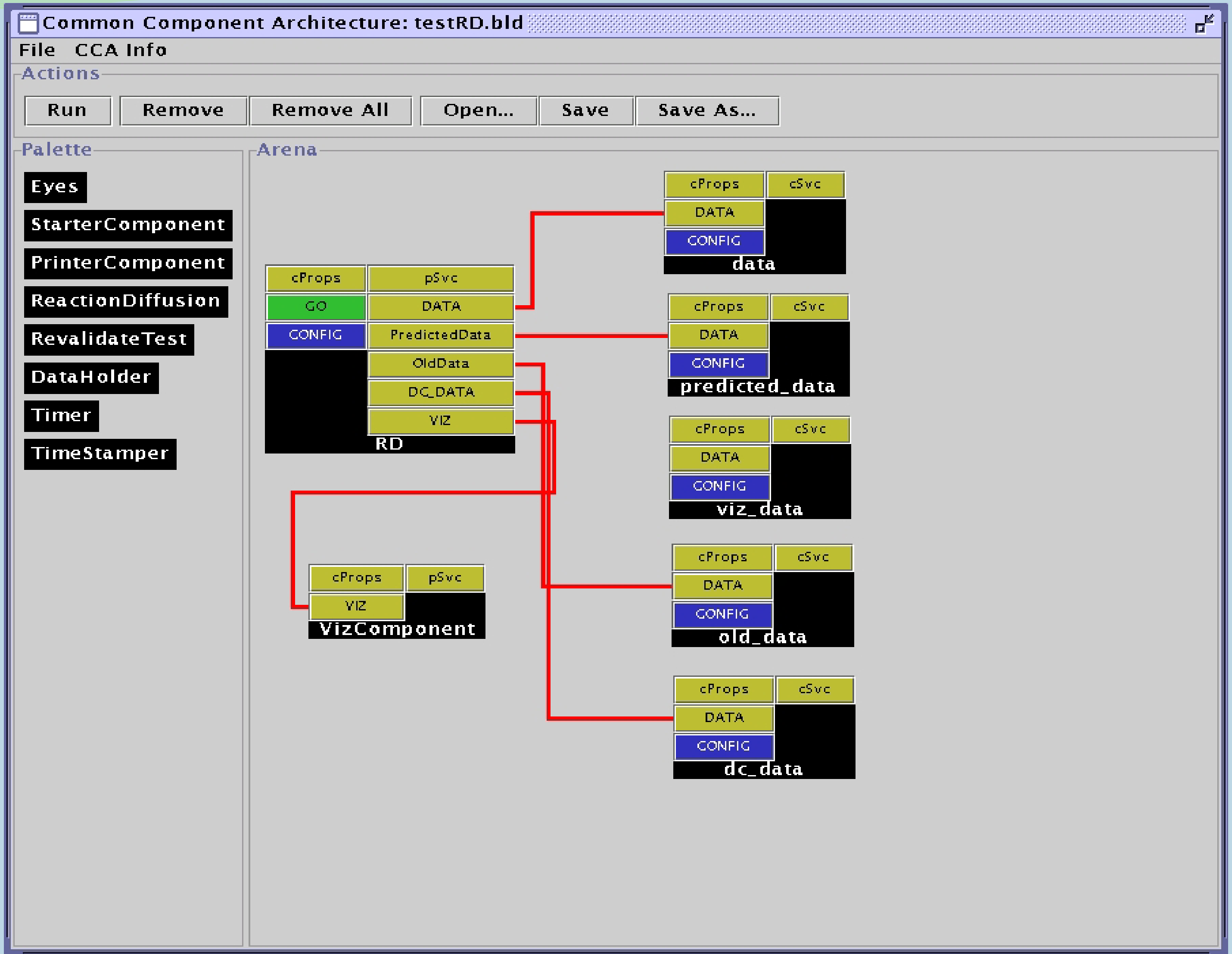
CCA Collective Port Modularizes Proc/Data Decomposition

Combining previous parallel component with a second parallel component in a different framework

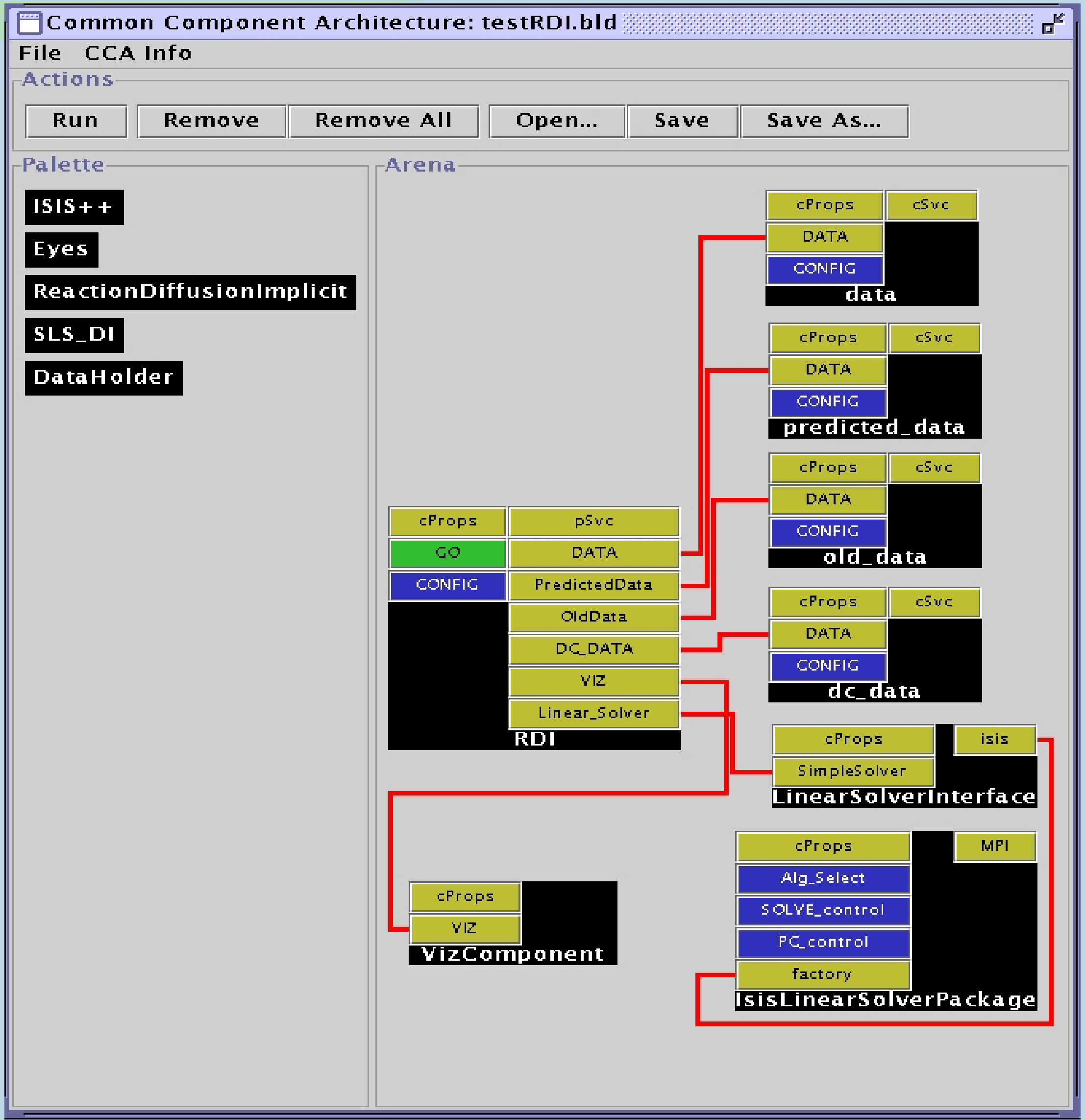


Container composed of ESI and Explicit Stencil components
parallel Viz component
collective port connecting M proc□s with N proc□s

A Simple example app



A little less simple



Its not that simple: detailed standards for HPC components are being worked out

- Scientific IDL
  -interfaces/lang. Interoperability for HP Computing
  - Scott Kohn (LLL: skohn@llnl.gov)
- Network-distributed components
  - line protocols, inter-framework compat. std's
  - Dennis Gannon (UI: gannon@cs.indiana.edu),
    Gary Kumfert (LLL: kumfert@esaki.llnl.gov)
- In-Memory Central Data Component
  -structured/unstructured/particle grid data std's
  - Lori Freitag(ANL: freitag@mcs.anl.gov)
- M processor to N Processor communication
  -reparitioning, inter-parallel program comm. std's
  - Jim Kohl (ORNL: kohlja@ornl.gov)

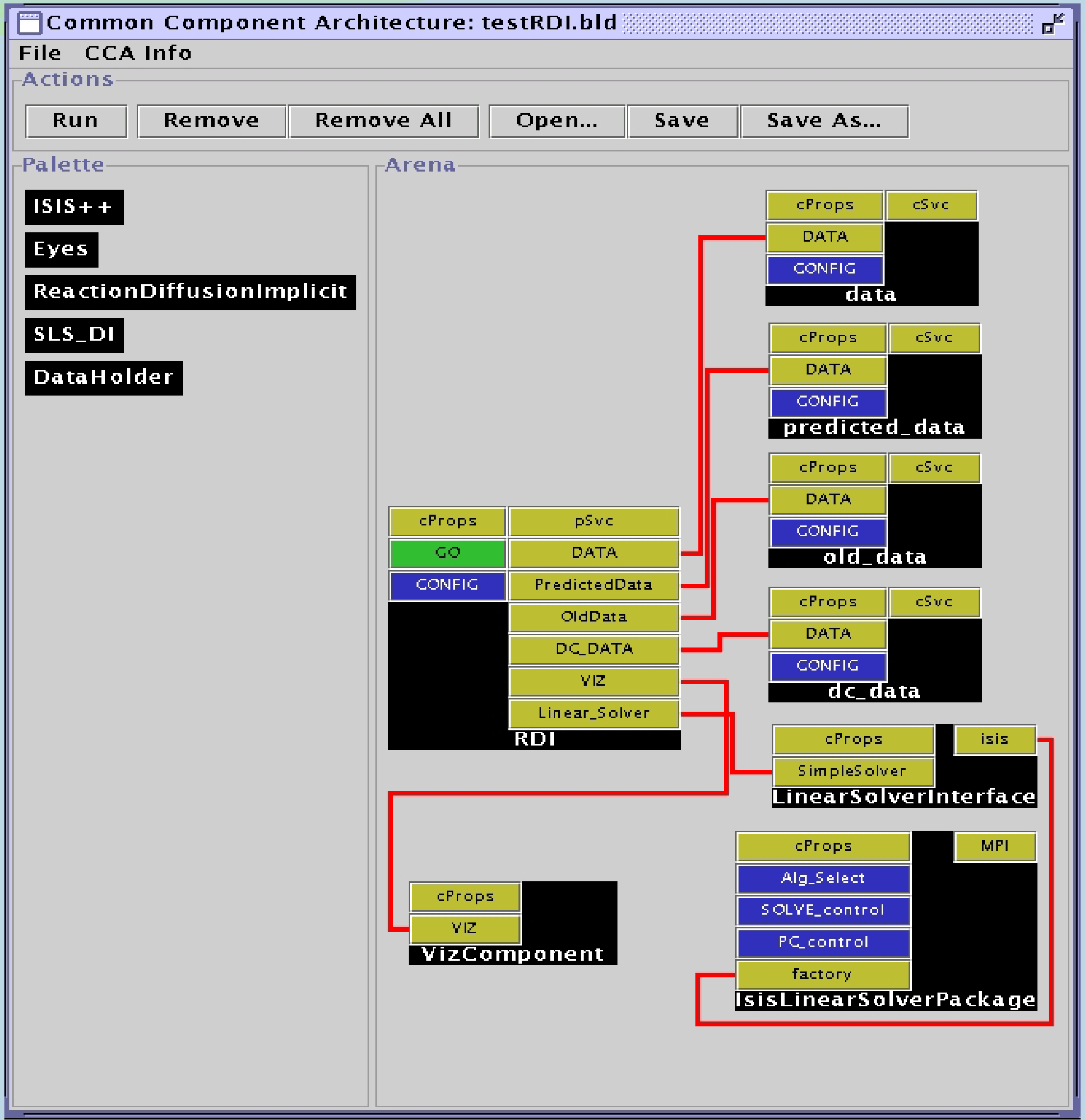
HPC Morphology: Big Central Data with numerical code modules that operate on it.

- Big Central Data Thing (BCDT) does R & T:
  - representation
    - proc. decomposed, present interface that abstracts data ...
  - transmission
    - proc. re-decomposition, interpolation, ...
- Numerical modules
  - intimately aware of BCDT, produce the answer
- CCA Version of these:
  - representation: freitag@mcs.anl.com
  - transmission: kohlja@ornl.gov

The devil in the nitty details ...

- "C++ is great as long as you stay away from ..."
  - templates, operator overloading?
    - we (Rob, maybe Ben & Jaideep) think that templates are part of the language (e.g.STL)
  - memory
    - force everyone to use your version of smart pointers?
- Using C++ arcanity in a standard will condemn everyone to using it.
  - recent controversy over exceptions in the CCA.
- SIDL will save us.

Python will save us ...

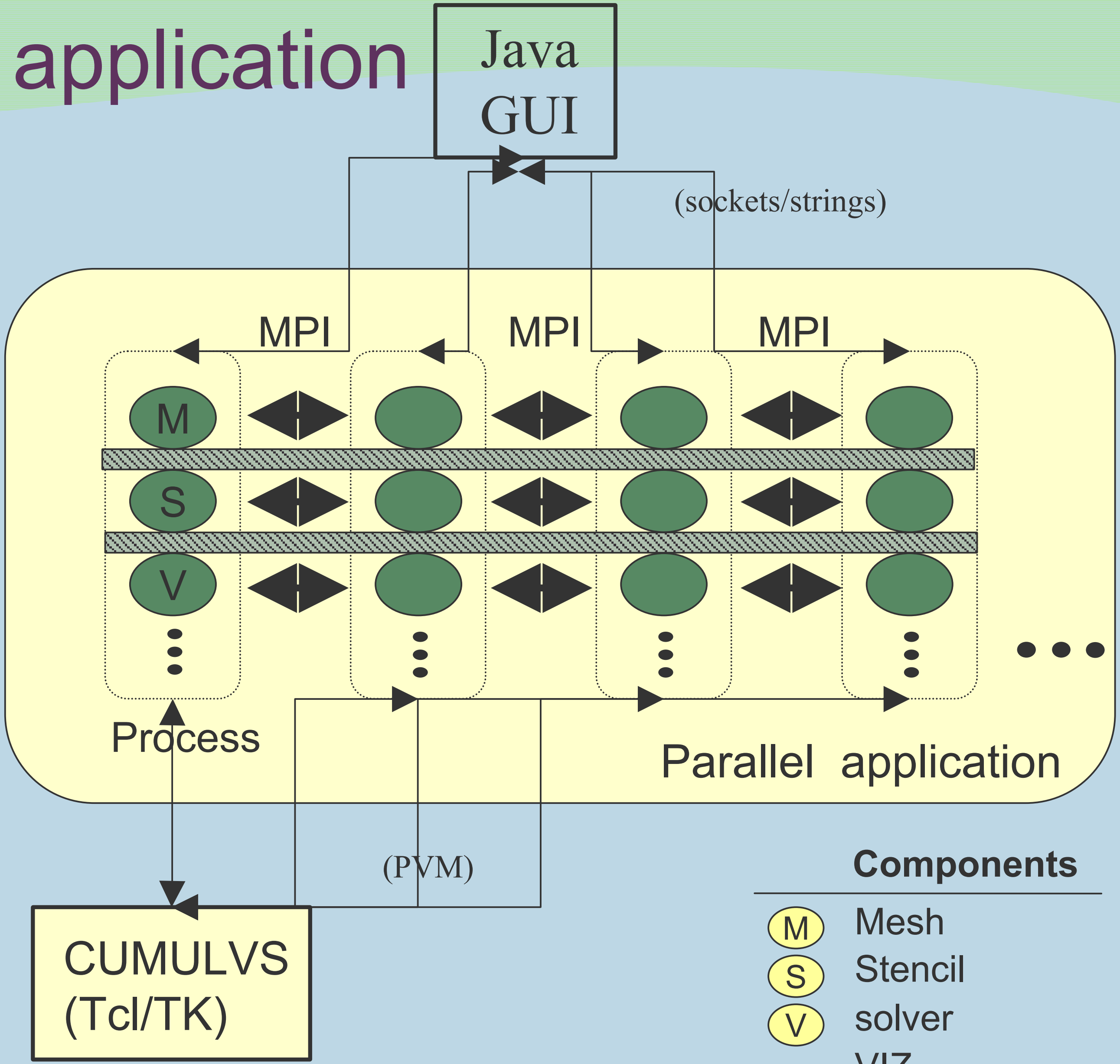


Where do you fit in?

- CCA does not pretend to be experts in all numerical algorithms, just provide a standard way to exchange component capability.
- Medium of exchange: interfaces
  - ESI has made headway toward defining interfaces for their realm
  - need numerics experts to define them for various fields other than ESI
    - optimization
    - nonlinear methods
    - symbolic and automatic methods
    - ?

Boneyard: End of talk

Sketch of HPC application



Components

M	Mesh
S	Stencil
V	solver
	VIZ

Scientific IDL provides OO interoperability to C, C++, Java, Python, Fortran 77/90

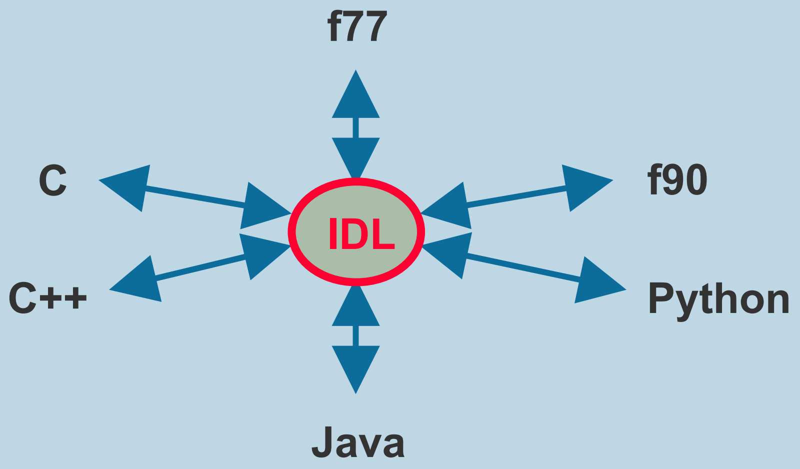
- Follows a Java-like interface model that will “glue” languages together with a minimum performance hit
  - expected overhead: 2-3 virtual function calls
- IDL interoperability library supports interface reflection
  - Necessary for run-time discovery of objects by frameworks
  - Allows interfaces to be “recognized” by the framework without having specific advanced knowledge of active components

Language interoperability is a critical first step towards software interoperability

- Software re-use is often hampered by language barriers
  - DOE labs use many languages (f77, f90, C, C++, Java, Python)
  - can be difficult for some languages to call others (f77 to C++)

- We are developing IDL technology for interoperability
  - interface definition language (IDL) describes calling interface
  - tools automatically generate code to □glue□ languages

A. Cleary, S. Kohn, S. Smith, B. Smolinski, *Language Interoperability Mechanisms for High-Performance Scientific Applications*, SIAM Interoperability Conference, 1998.



IDL specification for the SMG structured multigrid preconditioner from *hypre*

```
package hypre {
   class stencil {
      static stencil NewStencil(in int dim, in int size);
      int SetStencilElement(in int index, in array<int> offset);
   };
   class grid {
      static grid NewGrid(in mpi_com com, in int dimension);
      int SetGridExtents(inout array<int> lower, array<int> upper);
   };
   class vector {
      static vector NewVector(in mpi_com, in grid g, in stencil s);
      int SetVectorBoxValues(/* long argument list omitted */);
   };
   class matrix { /* matrix member functions omitted */ };
   class smg_solver {
      int Setup(inout matrix A, inout vector b, inout vector x);
      int Solve(inout matrix A, inout vector b, inout vector x);
   };
);
```

User applications can now invoke *hypre* routines from both C and Fortran 77

C Test Code

```
hypre_vector b, x;
hypre_matrix A;
hypre_smg_solver smg_solver;

b = hypre_vector_NewVector(com, grid, stencil);
...
x = hypre_vector_NewVector(com, grid, stencil);
...
A = hypre_matrix_NewMatrix(com, grid, stencil);
...

smg_solver = hypre_smg_solver_new();
hypre_smg_solver_SetMaxIter(smg_solver, 10);
hypre_smg_solver_Solve(smg_solver, &A, &b, &x);
hypre_smg_solver_Finalize(smg_solver);
```

Fortran 77 Test Code

```
integer b, x
integer A
integer smg_solver

b = hypre_vector_NewVector(com, grid, stencil)
...
x = hypre_vector_NewVector(com, grid, stencil)
...
A = hypre_matrix_NewMatrix(com, grid, stencil)
...

smg_solver = hypre_smg_solver_new()
call hypre_smg_solver_SetMaxIter(smg_solver, 10)
call hypre_smg_solver_Solve(smg_solver, A, b, x)
call hypre_smg_solver_Finalize(smg_solver)
```

The end user is completely unaware that IDL tools were used

Where are we?
Details of the CCA specification ...

- Overall: `http://www.cca-forum.org`
- draft Port spec in beta stage:
  `http://www.cca-forum.org/port-spec`
- draft IDL spec in RFC stage:
  `http://www.llnl.gov/CASC/babel`
- CCA SCMD Demo applet & plug-in:

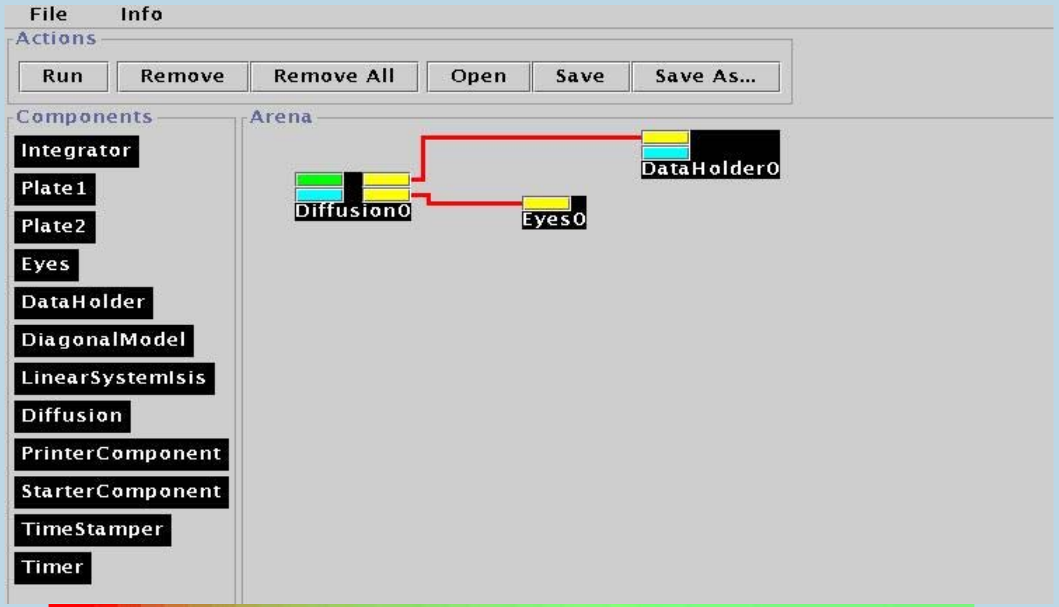
  `coming soon ...`
- Multiple test codes already exist.

Where are we going?

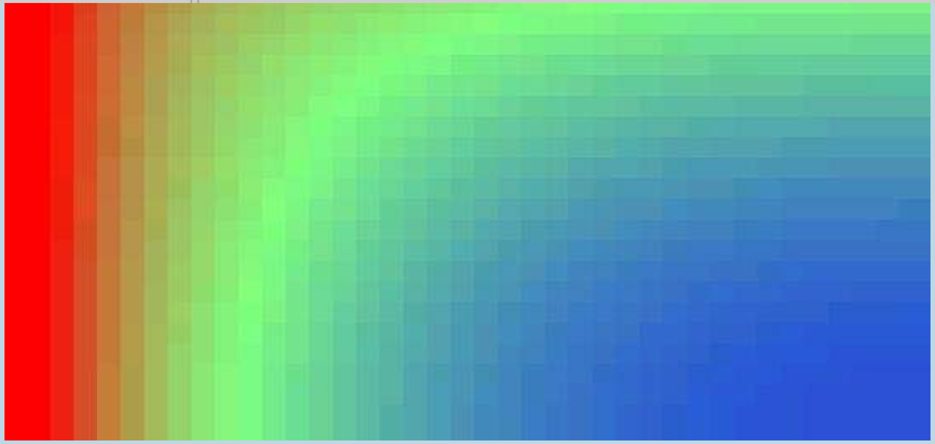
- Producing components and interface standards that people want to use:
  - Eqn solver (ESI)
  - MxN redecomposition interface and implementation
  - Data Decomp./Load balancing components
  - Component parameters
  - Component documentation
  - Component constraints/hints for data decomposition
    - Other domain-specific components and interfaces

- Experiment with the architecture.

CCA Reference Framework (SPMD Components Exist)



Framework GUI running on 9 Processors CPlant

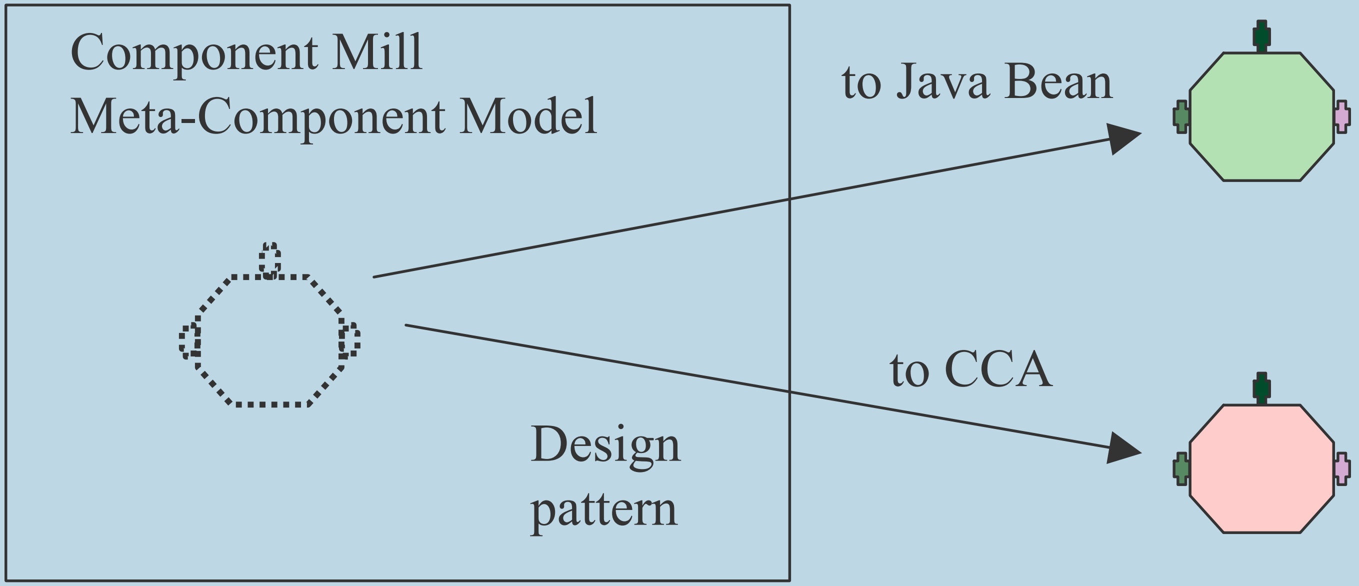


Visualization

CCA Distributed Object example

Bumping up the level of component abstraction

- Meta-components
  - viewed only in terms of the mechanisms for component composition.
  - component connection mechanisms abstracted



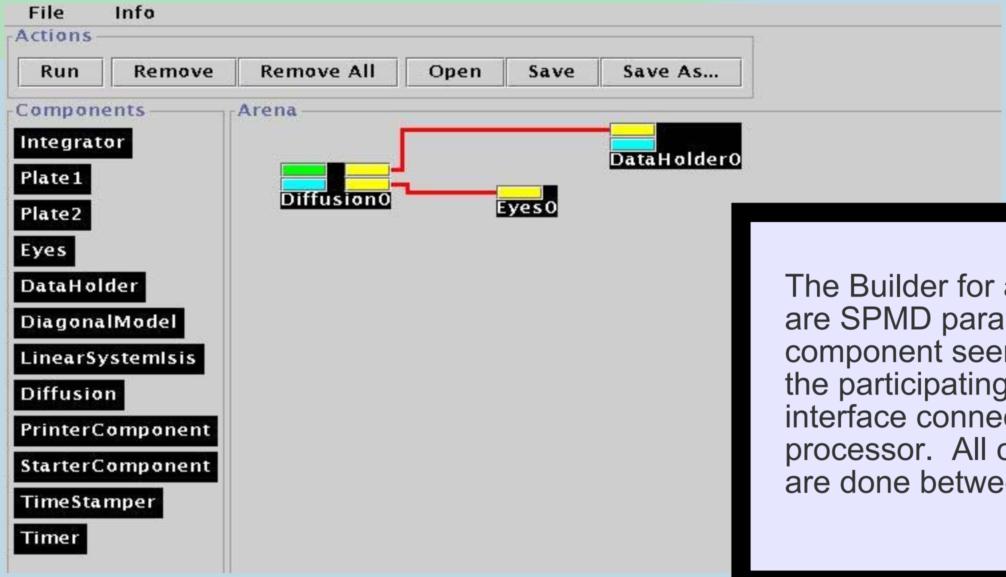
Metacomponents free the programmer from a specific Component Architecure

- Programmer is not strapped to a specific CA
  - Just needs to provide an *adaptor* to a CA supported by ASIA.
  - *Allows design decisions to be put off until deployment*
- *Does not free the programmer from user requirements*
  - *Need to understand the user domain*
  - *Need to understand where you fit in the big picture*
  - *Need to understand ASIA mechanics.*

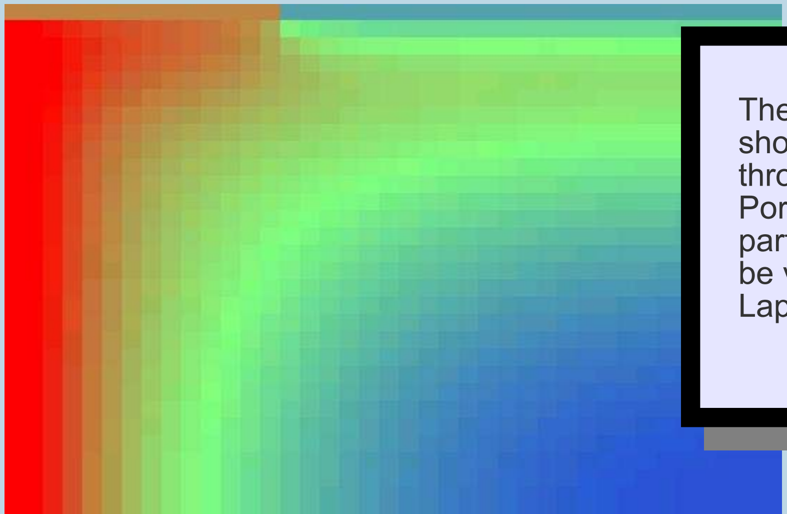
Scientific IDL provides OO interoperability to C, C++, Java, Python, Fortran 77/90

- Follows a Java-like interface model that will □glue□ languages together with a minimum performance hit
  - expected overhead: ~2-3 virtual function calls
- IDL interoperability library supports interface introspection
  - Necessary for run-time discovery of objects by frameworks
- **`http://www.llnl.gov/CASC/babel`**

CCA compliant VPL that runs in a web browser in full-up SPMD Fashion

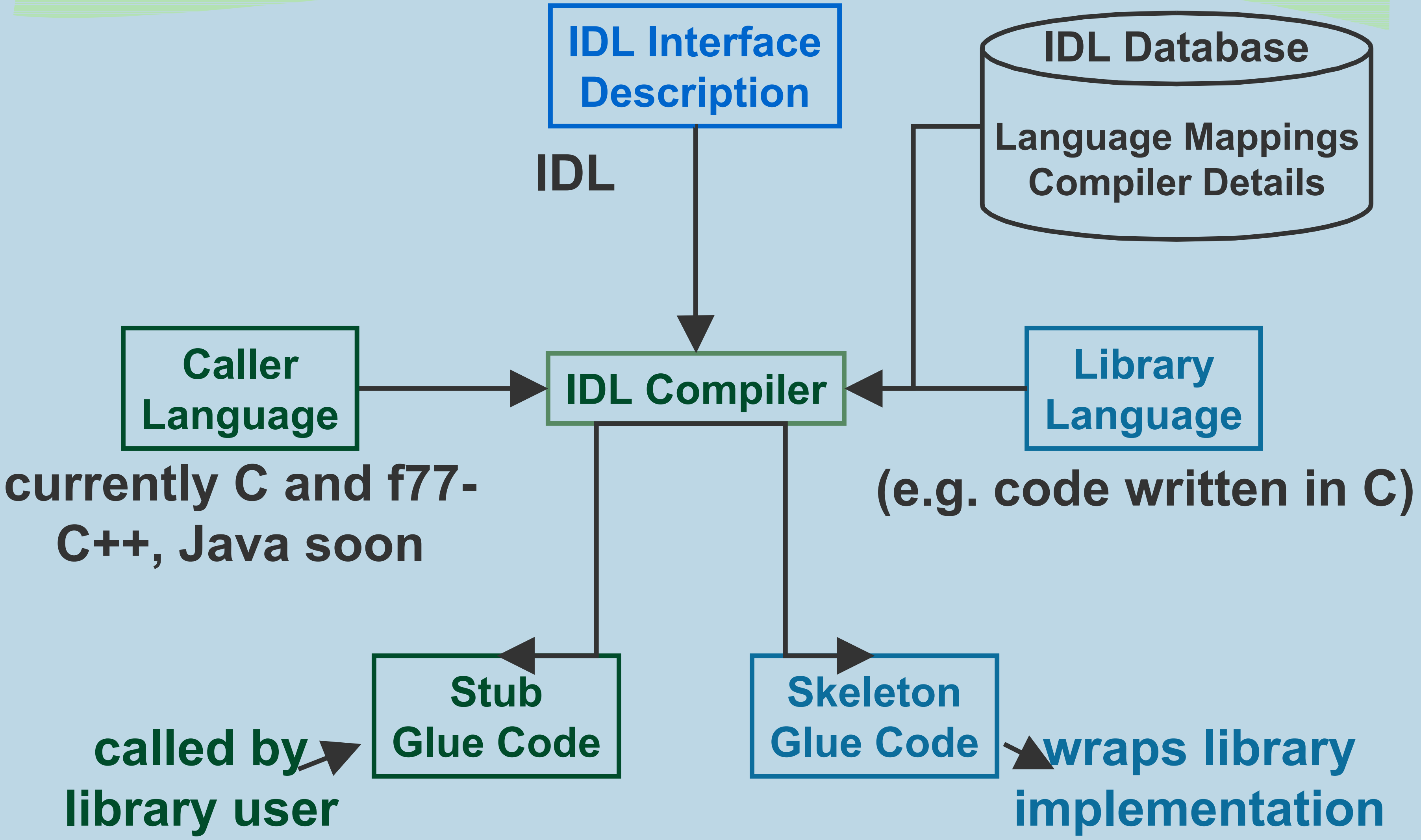


The Builder for a CCA reference implementation. These are SPMD parallel CCA 1.0 compliant components. Each component seen in the Builder is replicated on each of the participating processors. The red lines represent interface connections between components *within* a processor. All communication necessary to the solution are done between peer components.



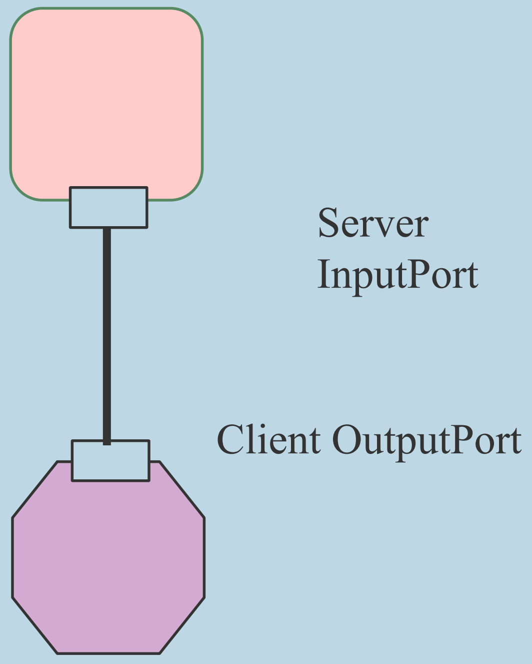
The visualization is done through the Eyes component shown above. This implements the CCA Collective Port through the Oak Ridge CUMULVS facility. The Collective Port's purpose is to assemble the partial data from the participating processors and send them to a single point to be visualized. The rendering here is the solution to Laplace's equation.

An IDL compiler automatically generates glue code for supported languages

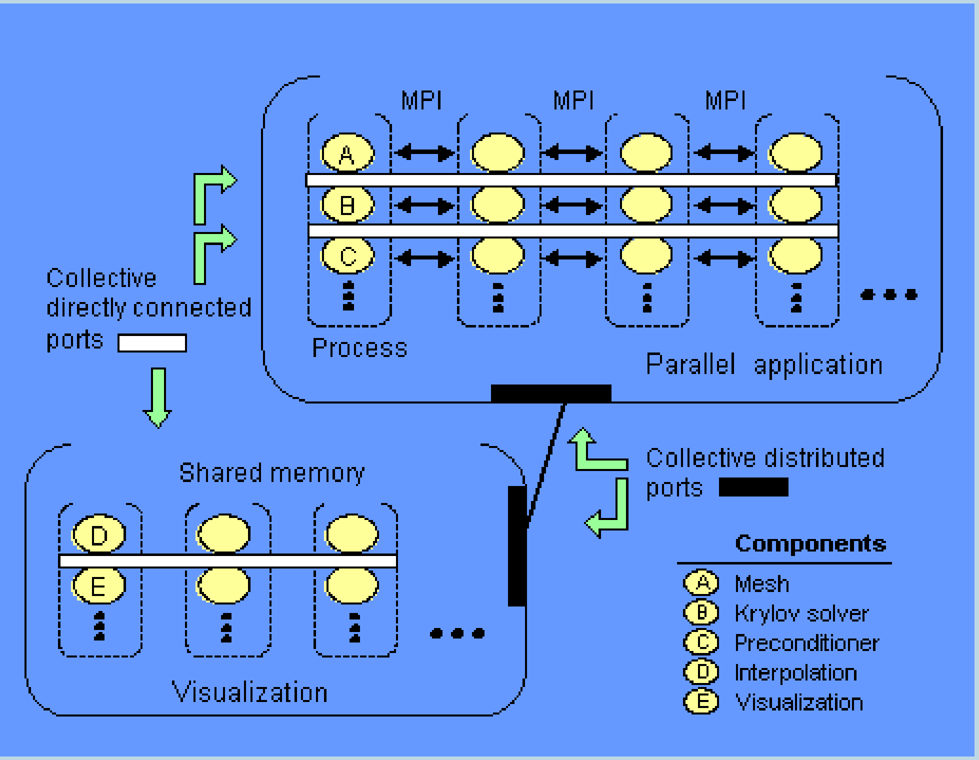


Ports can be used to characterize potential links to other components

- Each CCA Component has
  - A list of input Ports where
    - An input Port defines an interface to services provided by the component.
  - A list of output Ports where
    - An output Port is the interface to a set of services required by the component.
  - Other self-identification features
    - allow component containers to probe the component properties, ParameterPort.

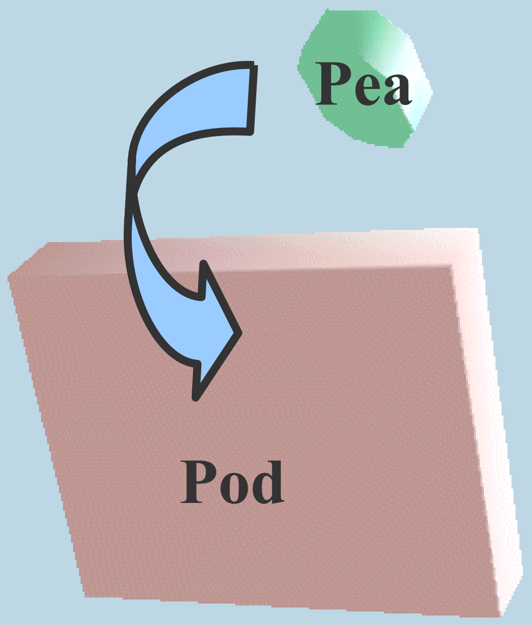


Where are we going?

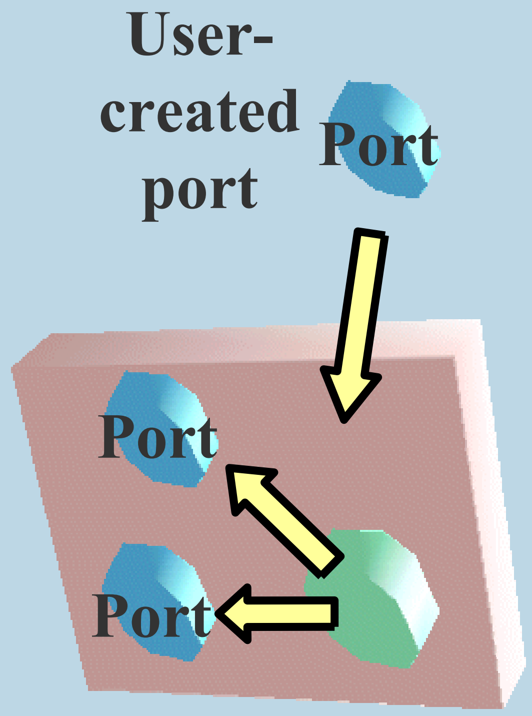


- Solving MxN problem
- Reference implementation of CCA with Equation Solver Interface ESI component.
- experiments with the architecture.

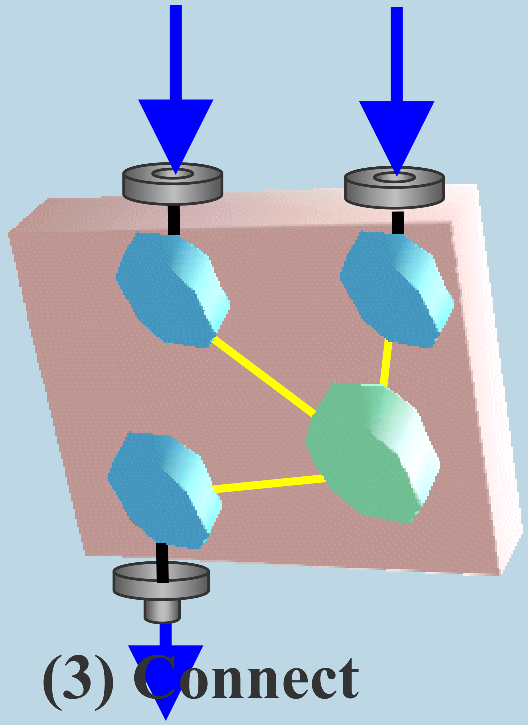
Programmer can create a new Port by tying methods to a Port



(1) Add pea to pod component



(2) Create input and output port objects via introspection



(3) Connect components

Conclusions

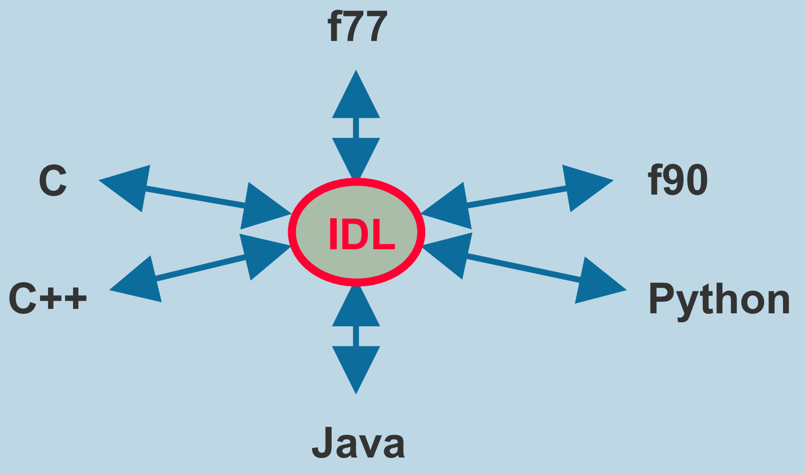
- Philosophy: Not trying to □own the world.□ Trying to provide a □coin-of-the-realm□ -- a medium of exchange -- that promotes an HPC software economy.
- Overall: `http://www.acl.lanl.gov/cca-forum`
- Draft Port spec in RFC stage: `http://z.ca.sandia.gov/~cca-forum/gport-spec`
- Draft IDL spec in RFC stage: `http://www.llnl.gov/CASC/babel`
- CCA Demo applet & plug-in:

  `http://z.ca.sandia.gov/~cca-forum/cca-demo`

Language interoperability is a critical first step towards software interoperability

- Software re-use is often hampered by language barriers
  - DOE labs use many languages (f77, f90, C, C++, Java, Python)
  - can be difficult for some languages to call others (f77 to C++)
- We are developing IDL technology for interoperability
  - interface definition language (IDL) describes calling interface
  - tools automatically generate code to glue languages

A. Cleary, S. Kohn, S. Smith, B. Smolinski, *Language Interoperability Mechanisms for High-Performance Scientific Applications*, SIAM Interoperability Conference, 1998.



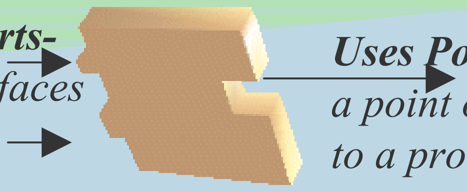
IDL specification for the SMG structured multigrid preconditioner from *hypre*

```
package hypre {
   class stencil {
      static stencil NewStencil(in int dim, in int size);
      int SetStencilElement(in int index, in array<int> offset);
   };
   class grid {
      static grid NewGrid(in mpi_com com, in int dimension);
      int SetGridExtents(inout array<int> lower, array<int> upper);
   };
   class vector {
      static vector NewVector(in mpi_com, in grid g, in stencil s);
      int SetVectorBoxValues(/* long argument list omitted */);
   };
   class matrix { /* matrix member functions omitted */ };
   class smg_solver {
      int Setup(inout matrix A, inout vector b, inout vector x);
      int Solve(inout matrix A, inout vector b, inout vector x);
   };
);
```

Common Component Architecture Toolkit

CCA Components

Provides Ports- *public interfaces to provided functions*

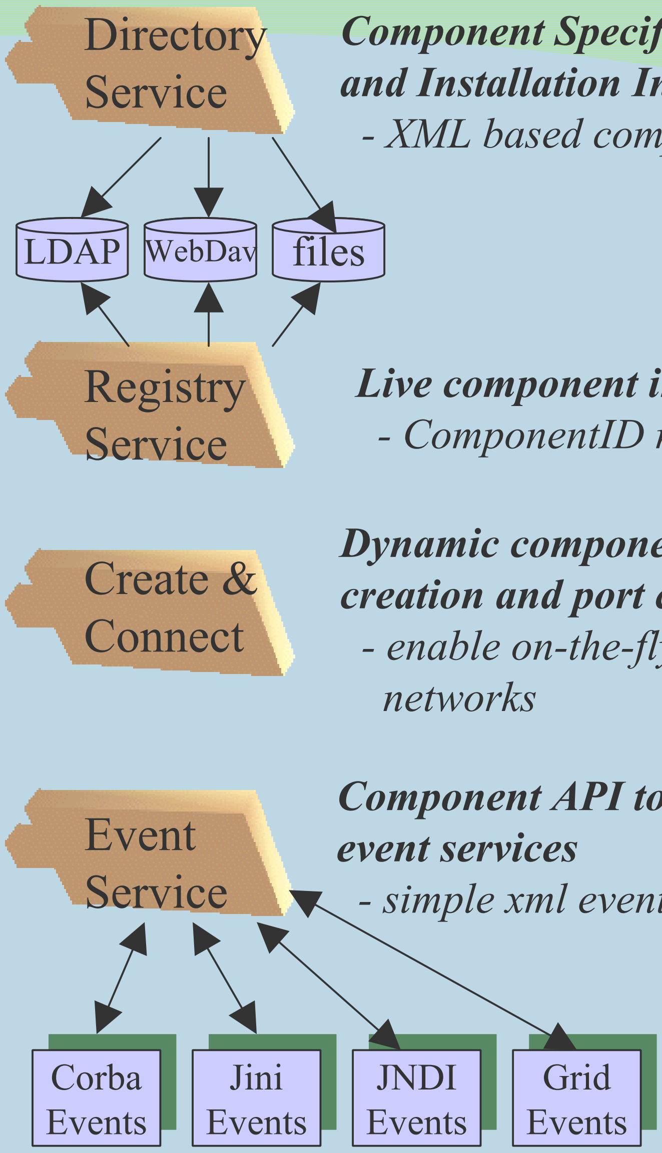


Uses Port- *a point of call to a provider*

A Component can be:
- A Desktop tool like Matlab or Python.
- A large remote parallel application
- An encapsulated remote instrument
- A data archive or database
- ...

CCAT Components are distributed and executed over wide area Grids.

CCAT Service Architecture



Component Specifications and Installation Info
- XML based component schema

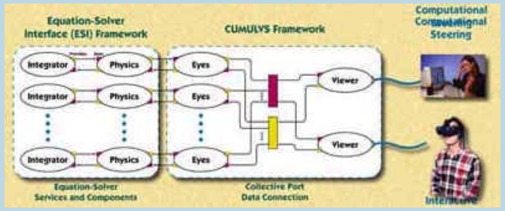
Live component instances
- ComponentID registry

Dynamic component instance creation and port connections
- enable on-the-fly and nested networks

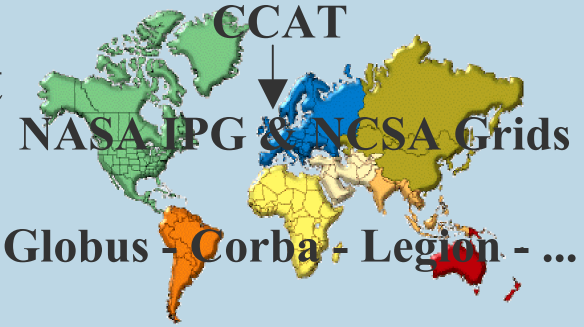
Component API to external event services
- simple xml event schema

Distributed Object Framework Using CCA

Equation Solver Interface framework and the Cumulvs Computational Steering framework connected using CCA ports.


Equation-Solver Interface (ESI) Framework
CUMULVS Framework
Computational Steering
Integrator
Physics
Eyes
Viewer
Equation-Solver Services and Components
Collective Port Data Connection

CCAT □ Indiana CCA Toolkit: including event service, component directory, dynamic creation and linking. Running over Grid resources



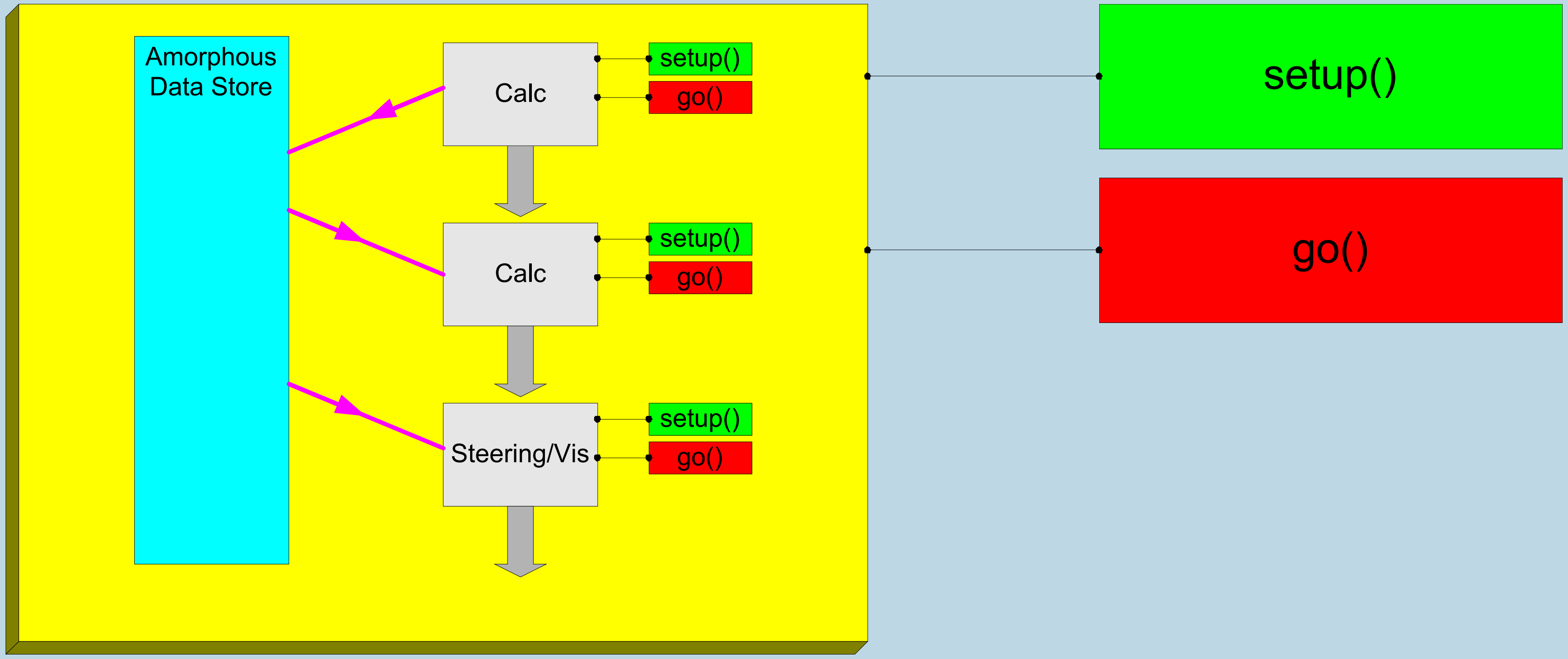
General Information: www.acl.lanl.gov/cca-forum
CCA spec in RFC stage, version 1.0 by Dec□99

http://z.ca.sandia.gov/~cca-forum/port-spec

Who are we?

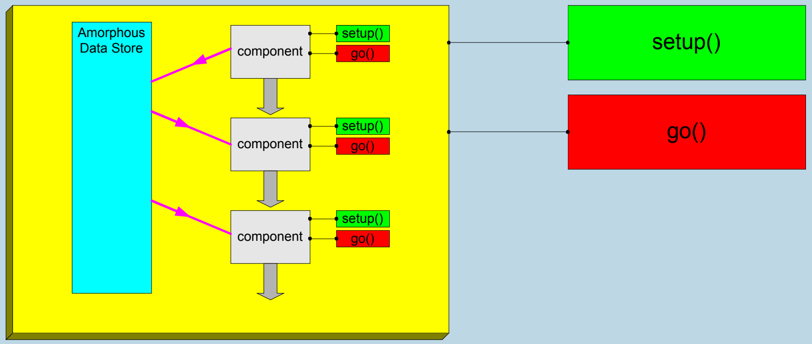
- Researchers in the HP components field that are dedicated to forming an open standard for HP components.
  - addressing the concerns of HPC.
- Originated from DOE□s DOE2K program
  - has its place in ASCI: Software Integration Curve.
  - participation from universities.

Container classes will orchestrate components data on which they depend

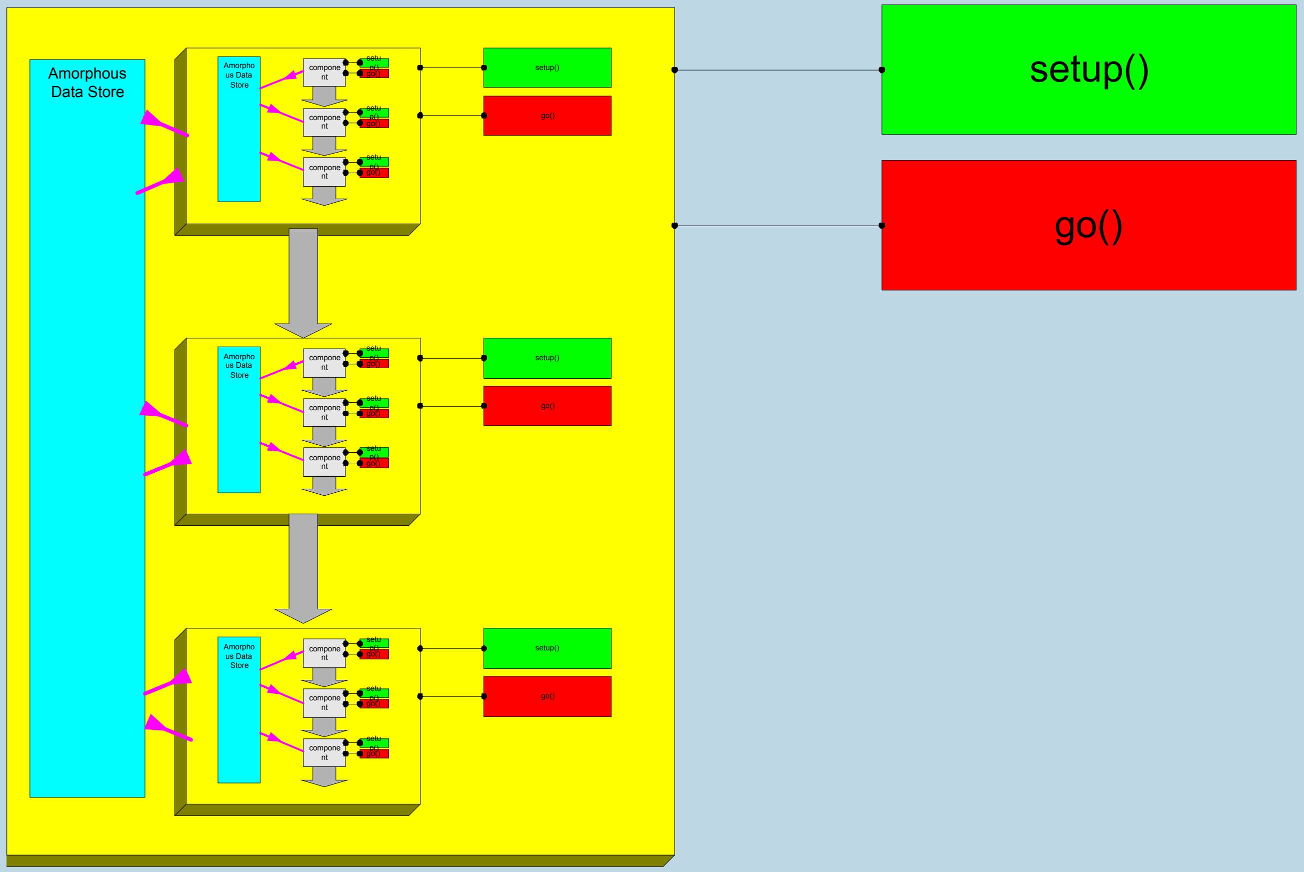


Container class gives access to common data and sequences components ...

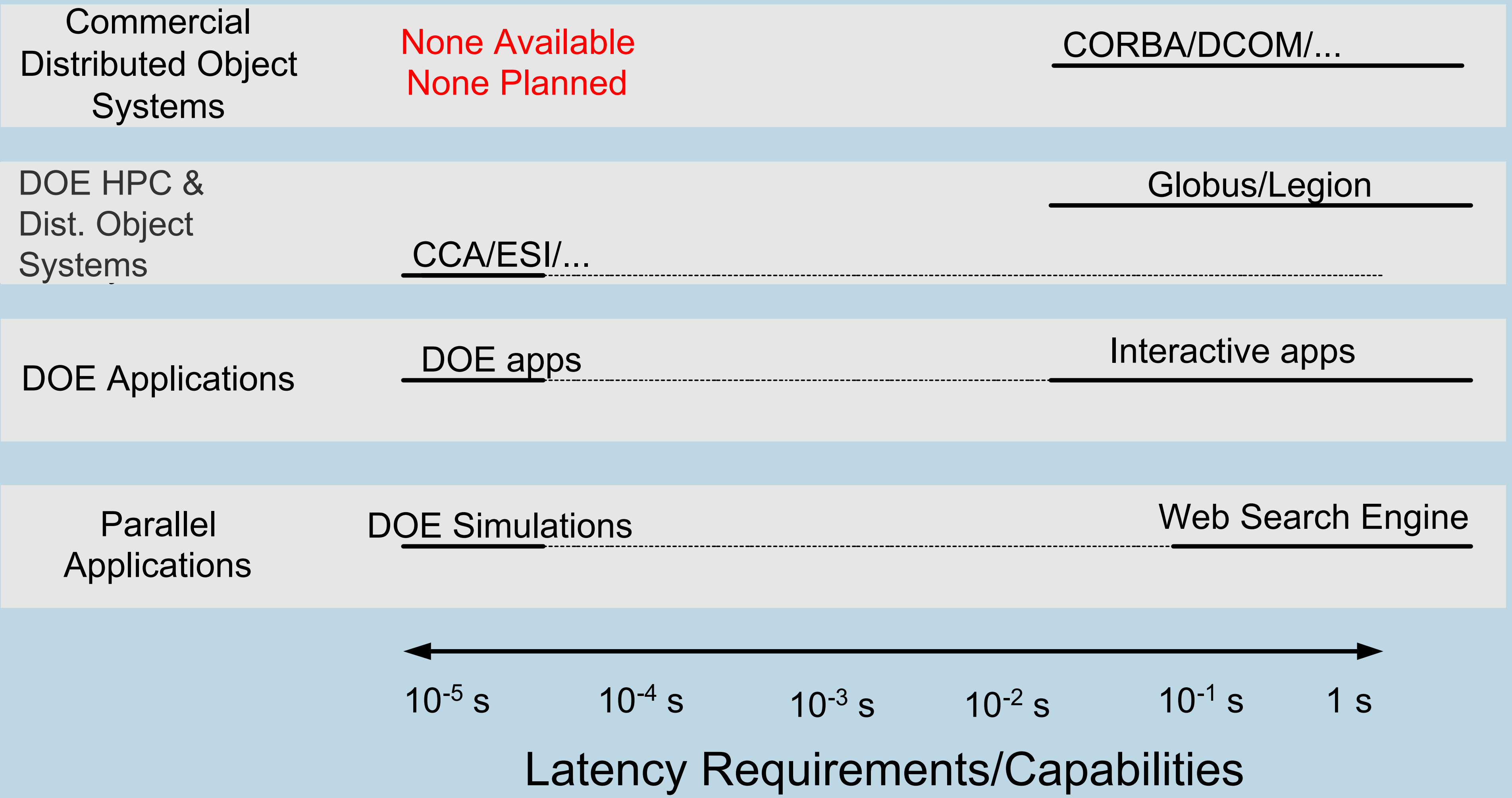
- Container is itself a component (complies with the object model).
  - Setup() means setup() its constituents.
  - Go() means run go() on its constituents.



Containers can have a data context …



HP Component Architecture meets a critical DOE need that *is not* and *will not* be addressed by Silicon Valley.



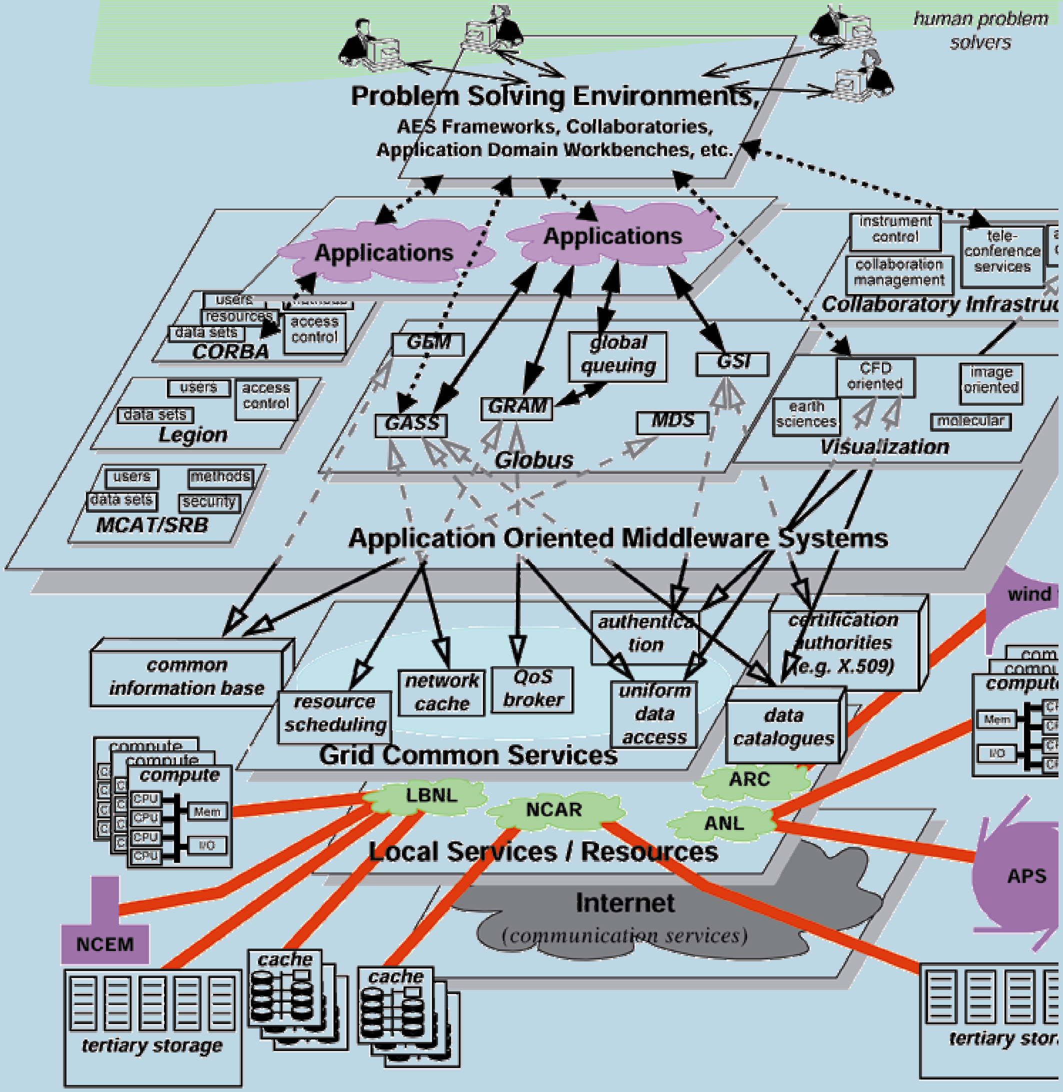
The problem is a lack of software integration.

- Designers and analysts will increasingly need access to complex computing tools and resources
- Computing resources are becoming more complex to manage
  - evolving systems present a moving target
  - heterogeneous, distributed computing is becoming the norm
  - portable, time-resistant solutions are needed
- Application components and tools are increasing in complexity
  - users need abstractions that partition their domain from other users
  - application requirements drive the need to move toward an inter-lab component-sharing model

The problem with existing Component Architectures

- Overall:
  - hi-latency connection is OK for loosely-coupled app□s: network OK for component composition
  - not OK for high-performance app□s.
- Specifics
  - JavaBeans: composition only by notifier/listener.
  - COM: no real interoperability, not vendor neutral.
  - CORBA2: no component model.
  - CORBA3 components: OK but complicated, only for loosely-coupled app□s.
- **Seek to supplement existing CA□s with an HP add-on.**

No plan for world domination...



- Not organizing complexity top-down.
- Provide an organizing principle (medium of exchange).
- believe in Adam Smith□s invisible hand ...